...e costruisci il tuo LABORATORIO DIGITALE

6,90 €

65

HARDWARE

DIGITALE DI BASE

MICROCONTROLLER

DIGITALE AVANZATO

TOTALMENTE PROGRAMMABILE!!!

9 771824 363008 41065

Peruzzo & C.

Direttore responsabile:
ALBERTO PERUZZO
Direttore Grandi Opere:
GIORGIO VERCELLINI
Consulenza tecnica e traduzioni:
CONSULCOMP S.n.c.
Pianificazione tecnica
LEONARDO PITTON

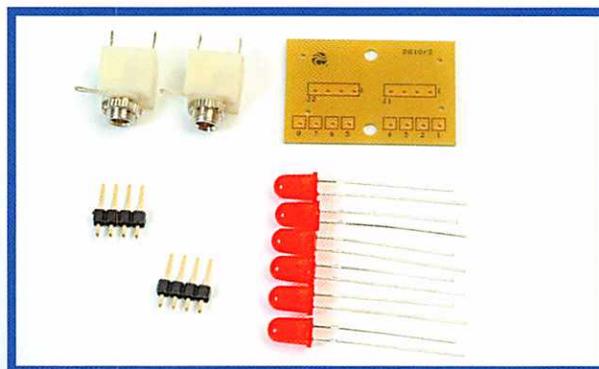
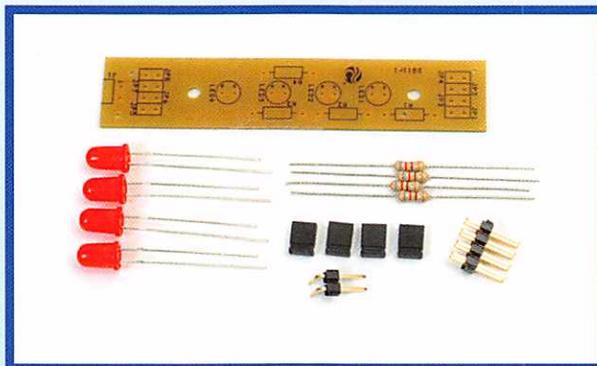
Direzione, Redazione, Amministrazione: viale Ercole Marelli 165, Tel. 02/242021, 20099 Sesto San Giovanni (Mi). Pubblicazione settimanale. Registrazione del Tribunale di Monza n. 1738 del 26/05/2004. Spedizione in abbonamento postale gr. II/70; autorizzazione delle Poste di Milano n. 163464 del 13/2/1963. Stampa: Grafiche Porpora s.r.l., Cernusco S/N (MI). Distribuzione SO.DI.P. S.p.A., Cinisello Balsamo (MI).

© 2004 F&G EDITORES, S.A.
© 2005 PERUZZO & C. s.r.l. Tutti i diritti sono riservati. Nessuna parte di questa pubblicazione può essere riprodotta, archiviata su sistema recuperabile o trasmessa, in ogni forma e con ogni mezzo, in mancanza di autorizzazione scritta della casa editrice. La casa editrice si riserva la facoltà di modificare il prezzo di copertina nel corso della pubblicazione, se costretta da mutate condizioni di mercato.

"ELETTRONICA DIGITALE"
si compone di
70 fascicoli settimanali
da suddividere
in 2 raccoglitori.

RICHIESTA DI NUMERI ARRETRATI. Per ulteriori informazioni, telefonare dal lunedì al venerdì ore 9.30-12.30 all'ufficio arretrati tel. 02/242021. Se vi mancano dei fascicoli o dei raccoglitori per completare l'opera, e non li trovate presso il vostro edicolante, potrete riceverli a domicilio rivolgendovi direttamente alla casa editrice. Basterà compilare e spedire un bollettino di conto corrente postale a PERUZZO & C. s.r.l., Ufficio Arretrati, viale Marelli 165, 20099 Sesto San Giovanni (MI). Il nostro numero di c/c postale è 42980201. L'importo da versare sarà pari al prezzo dei fascicoli o dei raccoglitori richiesti, più le spese di spedizione € 3,10 per pacco. Qualora il numero dei fascicoli o dei raccoglitori sia tale da superare il prezzo globale di € 25,82 e non superiore a € 51,65, l'invio avverrà per pacco assicurato e le spese di spedizione ammonteranno a € 6,20. La spesa sarà di € 9,81 da € 51,65 a € 103,29; di € 12,39 da € 103,29 a € 154,94; di € 14,98 da € 154,94 a € 206,58; di € 16,53 da € 206,58 in su. Attenzione: ai fascicoli arretrati, trascorse dodici settimane dalla loro distribuzione in edicola, viene applicato un sovrapprezzo di € 0,52, che andrà pertanto aggiunto all'importo da pagare. Non vengono effettuate spedizioni contrassegno. Gli arretrati di fascicoli e raccoglitori saranno disponibili per un anno dal completamento dell'opera. IMPORTANTE: è assolutamente necessario specificare sul bollettino di c/c postale, nello spazio riservato alla causale del versamento, il titolo dell'opera nonché il numero dei fascicoli e dei raccoglitori che volete ricevere.

IN REGALO in questo fascicolo

1 Scheda DG10
2 Jack femmina da pannello
2 Connettori da c.s. maschio, diritti a 4 vie
6 LED rossi

IN REGALO nel prossimo fascicolo

1 Scheda DG11r1
4 LED rossi
1 Connettore da c.s. maschio, diritto, da 2x4 vie
1 Connettore da c.s. maschio, a 90°, a 2 vie
6 Resistenze da 820 Ohm 5% 1/4 W
4 Ponticelli isolati neri

COME RACCOGLIERE E SUDDIVIDERE L'OPERA NELLE 4 SEZIONI

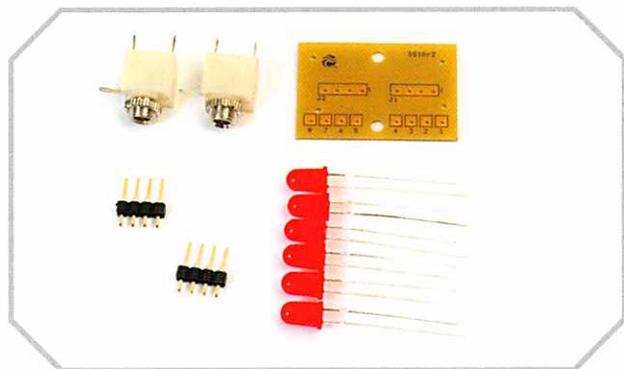
L'Opera è composta da 4 sezioni identificabili dalle fasce colorate, come indicato sotto. Le schede di ciascun fascicolo andranno suddivise nelle sezioni indicate e raccolte nell'apposito raccoglitore, che troverai presto in edicola. Per il momento, ti consigliamo di suddividere le sezioni in altrettante cartellette, in attesa di poterle collocare nel raccoglitore. A prima vista, alcuni numeri di pagina ti potranno sembrare ripetuti o sbagliati. Non è così: ciascuno fa parte di sezioni differenti e rispecchia l'ordine secondo cui raccogliere le schede. **Per eventuali domande di tipo tecnico scrivere al seguente indirizzo e-mail:** elettronicadigitale@microrobots.it

Hardware Montaggio e prove del laboratorio

Digitale di base Esercizi con i circuiti digitali

Digitale avanzato Esercizi con i circuiti sequenziali

Microcontroller Esercizi con i microcontroller

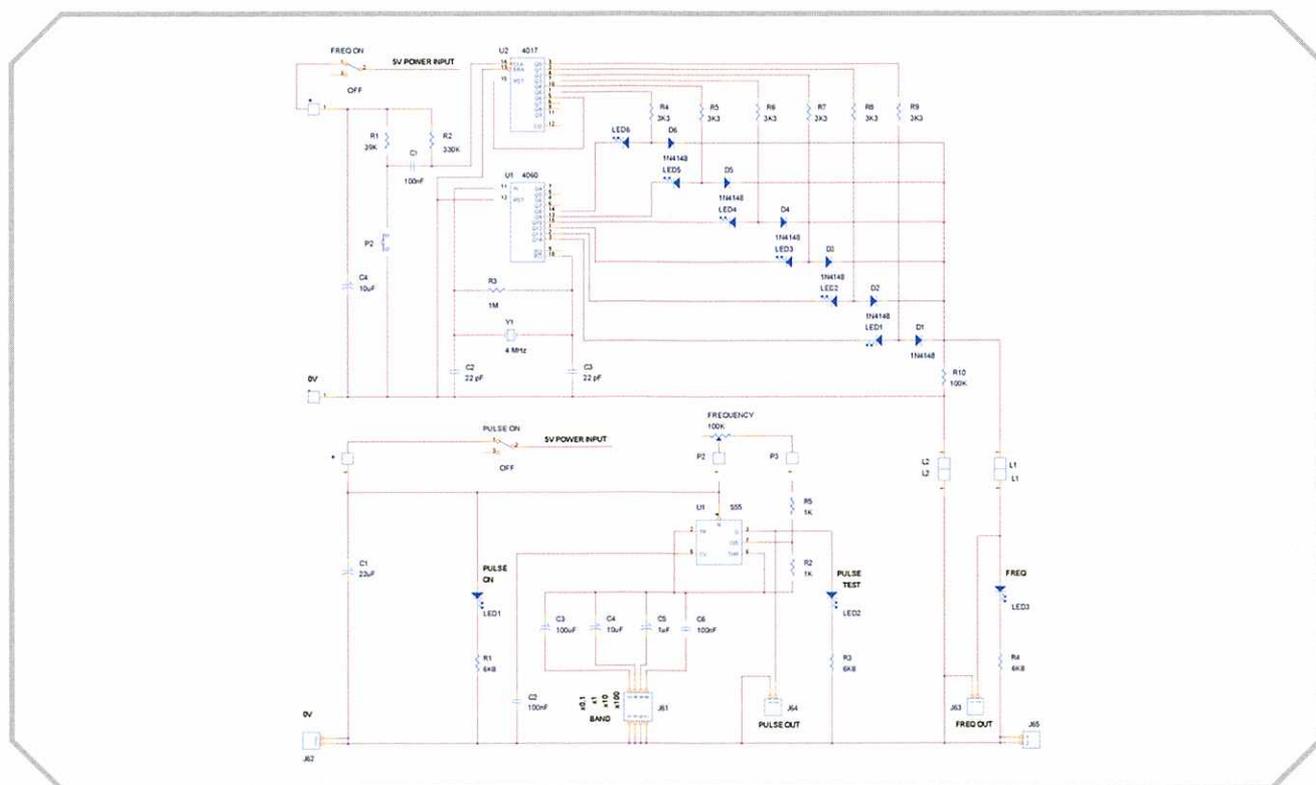
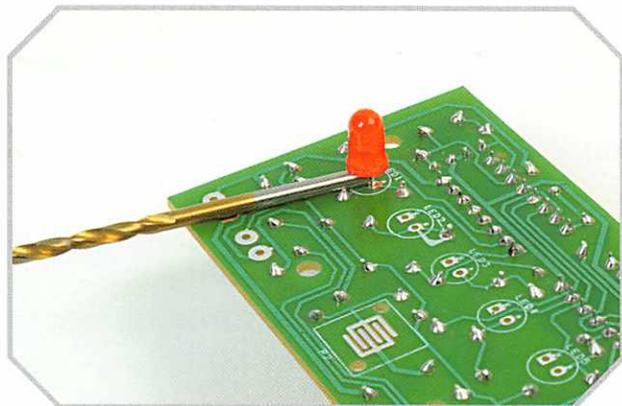
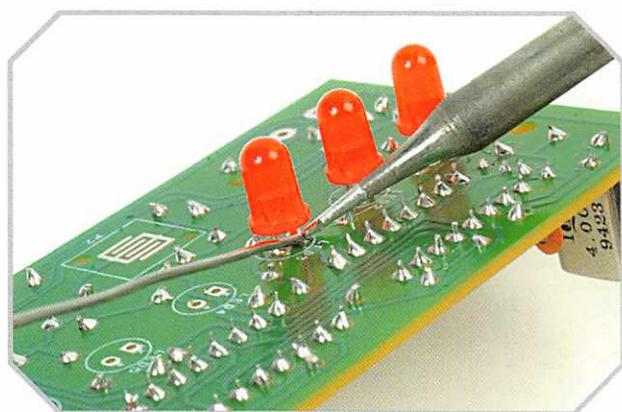
Generatore al quarzo (III)

Componenti allegati a questo fascicolo.

Dettaglio del montaggio di un LED.

Allegati a questo fascicolo troverete i sei LED necessari per terminare il montaggio della scheda del generatore DG17 oltre a tutti i componenti necessari per montare la scheda DG10. Il montaggio di quest'ultima verrà spiegato nel prossimo numero.

I LED

I sei LED contenuti in questo circuito stampato verranno inseriti dal lato saldature, come possiamo vedere nelle fotografie. Il LED è un componente che ha polarità, il terminale del catodo si identifica perché è il più vicino alla zona piatta della base del componente, sulla serigrafia è indicato con la lettera K.

La base del contenitore del LED deve rimanere separata all'incirca di 2,5 mm dal circuito stampato, in modo che possa fuoriuscire dai

Schema elettrico.

I LED devono essere separati di circa 2,5 mm dalla scheda.

Dettaglio delle saldature dei LED.

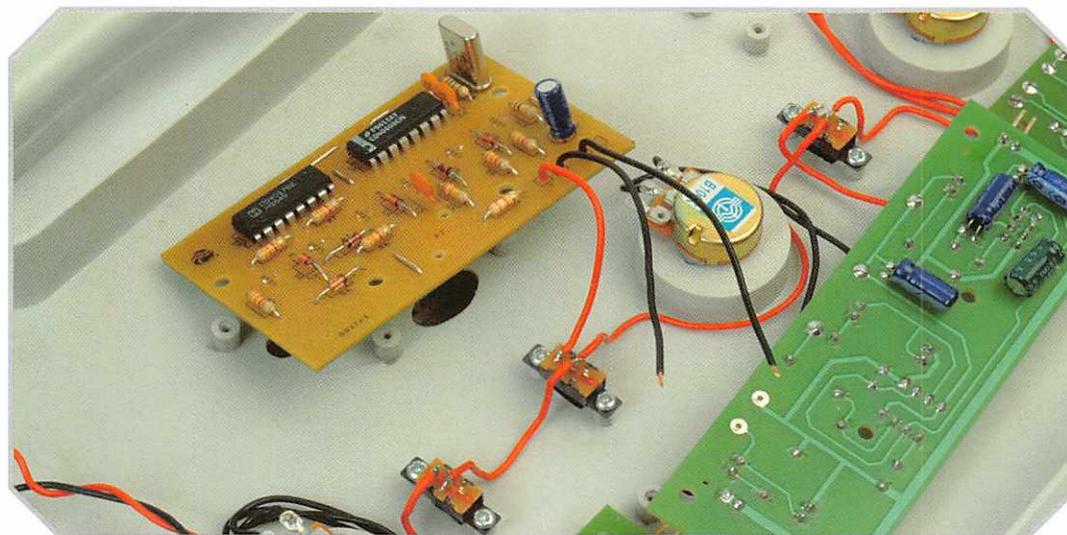
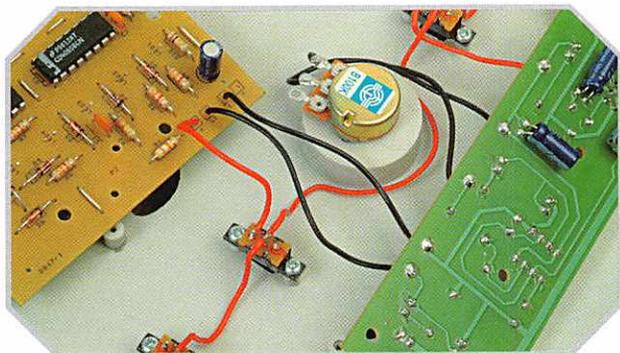
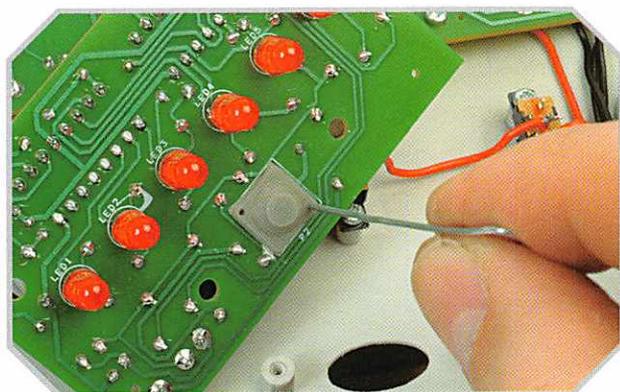
fori del pannello superiore, identificati da F1 a F6. Questa distanza permette anche di eseguire con una certa facilità le saldature, per mantenere la distanza del LED dalla scheda vi consigliamo di utilizzare una punta da trapano di 2,5 mm di diametro o qualche oggetto simile, in modo che tutti i LED rimangano alla stessa altezza. Dobbiamo fare attenzione alla polarità di ogni LED che inseriamo.

Collegamenti interni

Questa scheda si collega alla DG16 con due fili di colore nero da 7,5 cm di lunghezza, unendo fra loro i terminali che hanno sigle uguali sulle schede, in questo caso L1 e L2. Osservando lo schema possiamo verificare che L2 stabilisce il collegamento del negativo dell'alimentazione. Per realizzare questo collegamento conviene smontare la scheda DG16 corrispondente al generatore di impulsi, e la DG15 che contiene l'amplificatore audio, anche se con un minimo di abilità è possibile eseguire le due saldature a L1 e L2 senza smontarle.

Il collegamento del positivo dell'alimentazione si esegue con un filo di colore rosso da 6 cm di lunghezza tra il terminale + della scheda DG17 e il commutatore FREQ ON. Questo commutatore che fornisce l'alimentazione a questa scheda.

Pulsante

Questa scheda contiene un pulsante identificato con P2 sulla scheda e come FREQ UP sul pan-

Collegamenti della scheda DG17.

Scheda collegata al laboratorio.

Pulsante di silicone.

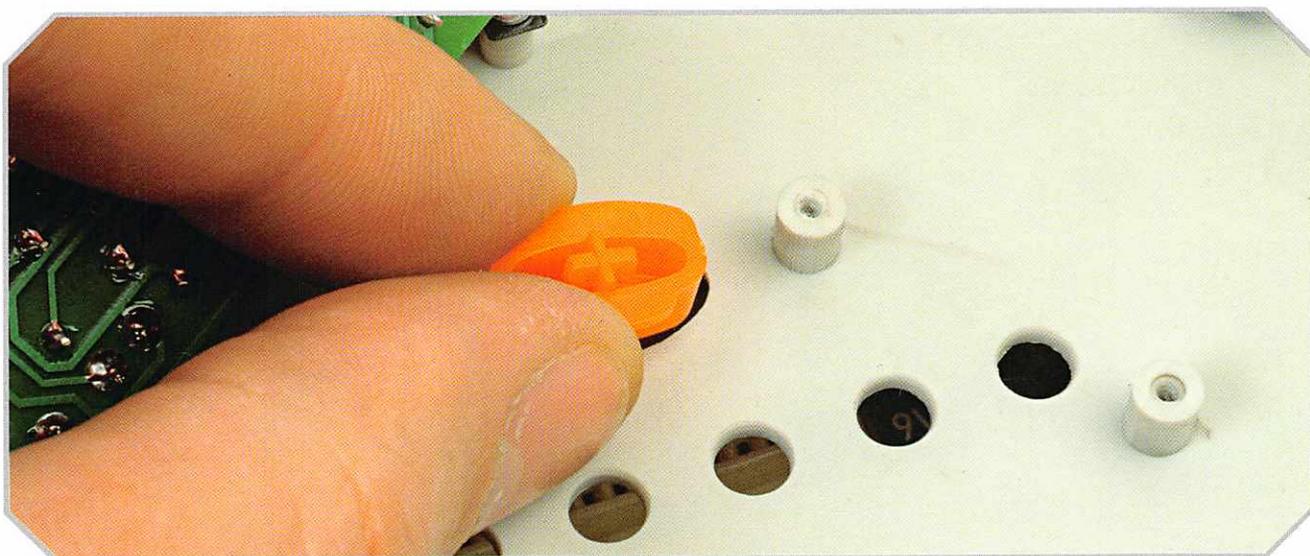
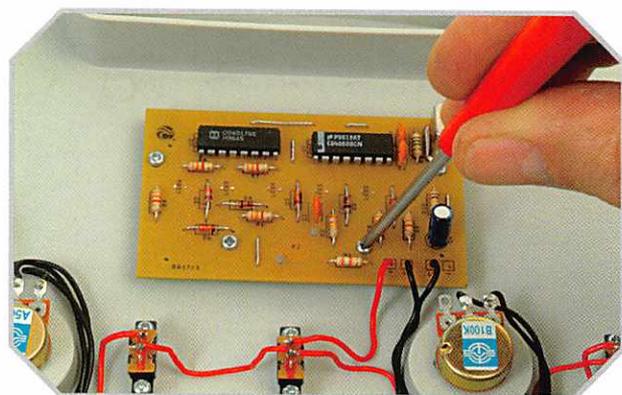
nello frontale, il pulsante propriamente detto vi è già stato fornito con il numero 40 e si monta dal lato saldature, inserendo le due guide dello stesso materiale del pulsante nei due fori da 2 mm di diametro; per aiutarci nell'operazione è sufficiente utilizzare una piccola clip da ufficio per fare entrare le due guide. L'attuatore del pulsante è un tasto di colore arancio che si inserisce dall'interno del pannello frontale nel foro a lui destinato, verificando in precedenza che non ci siano sbavature di plastica che ne impediscano il libero movimento.

Installazione della scheda

Per montare la scheda bisogna liberare gli alloggiamenti delle viti e inserirla facendo attenzione che il pulsante rimanga di fronte al tasto e che i sei LED fuoriescano dal pannello frontale. Dopo aver eseguito questa operazione si fissa la scheda con quattro viti, che vi sono già state fornite, ricordando di stringerle leggermente per non rovinare il filetto che le viti stesse creano nella colonna di plastica in cui vengono avvitate.

Se sono state tolte altre schede per saldare i fili, le dobbiamo fissare nuovamente con le loro viti.

Collegamenti esterni

L'uscita di questa scheda passa attraverso il connettore FREQ OUT posizionato sul pannel-

Montaggio del tasto di plastica.

Scheda nella sua posizione.

Generatore di frequenze.

lo superiore del laboratorio. Il segnale esce dal terminale siglato con un punto rosso, e la massa, che va collegata al negativo dell'alimentazione, è disponibile sul terminale identificato con un punto nero.

Utilizzo

L'oscillatore al quarzo si pone in funzione collegando il commutatore FREQ sulla posizione ON. Ci sono sei possibili frequenze disponibili, identificate come F1, F2, F3, F4, F5 e F6 la frequenza disponibile sul connettore FREQ OUT corrisponde a quella del LED illuminato.

Per passare da una frequenza alla successiva si preme FREQ UP, e arrivati all'ultima frequenza passa alla prima con un altro impulso. Possiamo vedere che il LED illuminato varia indicando quale frequenza è disponibile sull'uscita.

Si ottengono le seguenti frequenze:

LED	Frequenza
F1	15.625,00 Hz
F2	7.812,50 Hz
F3	3.906,25 Hz
F4	976,50 Hz
F5	488,00 Hz
F6	244,10 Hz

Pannello superiore del laboratorio.

Prova del generatore al quarzo

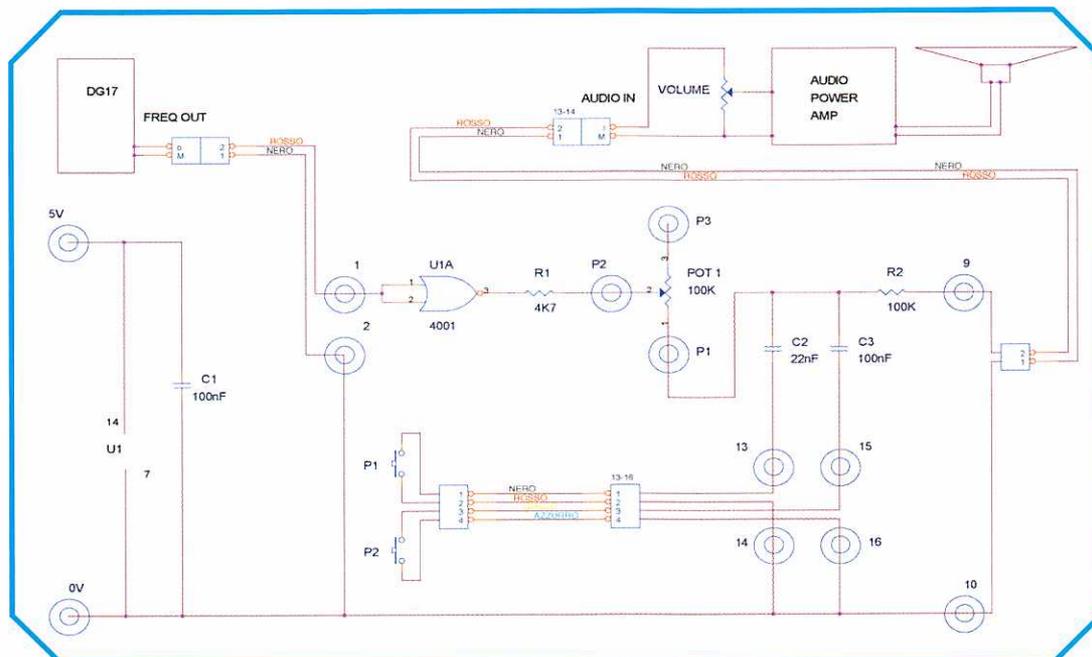
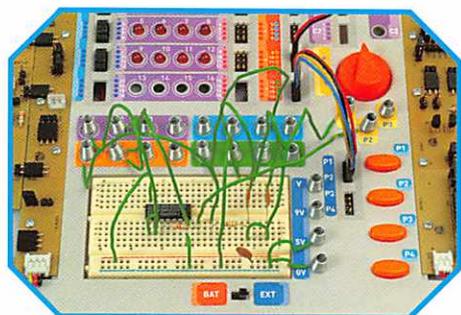
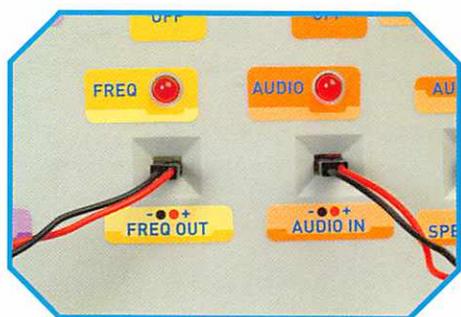
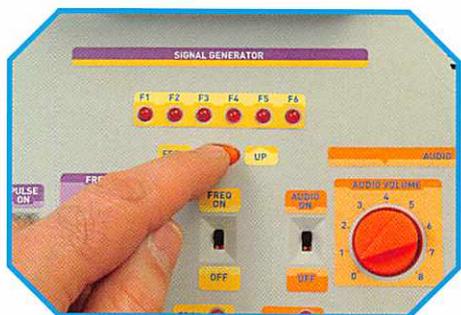
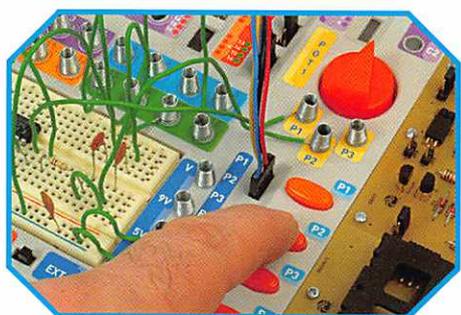
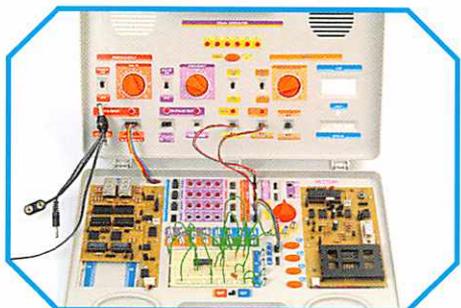
Con questo semplice circuito si prova il generatore di frequenze fisse del laboratorio. Questo circuito, anche se non misura la frequenza, ci permette di osservarne i cambiamenti, dato che le frequenze generate sono tutte udibili.

Il circuito

La porta U1A collegata all'uscita del circuito del generatore al quarzo si utilizza per avere a disposizione un segnale con i livelli logici da 0 a 5 volt, dato che il segnale dell'uscita diretta del generatore risente della caduta di tensione sui diodi e questo potrebbe rappresentare un problema per alcuni tipi di circuiti. Aggiungendo questa porta si ottiene un'uscita con dei livelli di tensione normalizzati per questo tipo di porte. Osservando l'uscita del circuito vediamo che premendo P1 o P2 si collega una resistenza in serie, formata da R1 e dal potenziometro, e un condensatore parallelo. Questi componenti formano un filtro passa-basso che ci permette di tagliare le frequenze più alte del segnale. Il circuito è progettato per fare delle prove.

Dobbiamo ricordare che un'onda quadra non è un tono puro, dato che, oltre al segnale della frequenza fondamentale f, contiene altre frequenze che corrispondono alle armoniche dispari superiori, ovvero : sf, 5f, 7f, ecc. L'utilizzo di un filtro passa-basso permette di attenuare in gran parte queste frequenze e l'udito può apprezzare importanti variazioni nel suono che sono molto facili da comparare, infatti basta rilasciare il pulsante che collega il condensatore e il filtro cessa di funzionare.

Componenti sulla scheda Bread Board.

Collegamenti della scheda.

Schema elettrico.

Collegamento dei pulsanti.

Uscita del generatore al quarzo.

Premendo FREQ UP si cambia la frequenza.

Premendo P1 o P2 si collega il filtro passabasso.

Esperimento completato.

Montaggio

Il montaggio di questo esercizio è rapido e si esegue come d'abitudine senza dimenticare di alimentare il circuito integrato. L'uscita del generatore al quarzo siglata come FREQ OUT si porta, mediante un filo a due conduttori, ai connettori corrispondenti alle molle 1 e 2, in modo che il filo rosso colleghi l'uscita FREQ OUT, punto rosso, con l'ingresso della porta U1A. L'uscita del circuito utilizza i terminali 9 e 10, che si collegano anch'essi con un cavo a due conduttori all'ingresso dell'amplificatore AUDIO IN. Il collegamento dei pulsanti si esegue con un cavetto a quattro conduttori utilizzando le connessioni delle molle dalla 13 alla 16.

Funzionamento

Dopo aver verificato tutti i collegamenti si alimenta il circuito con un alimentatore esterno o con le batterie. In quest'ultimo caso sarà sufficiente il primo portabatterie, posizionando il commutatore dell'alimentazione sul pannello principale nella posizione BAT; nel caso in cui si utilizzi l'alimentazione della rete è necessario collegare l'alimentatore e impostare il commutatore anteriore su EXT e quello del POWER su ON, oltre a portare su ON quello di FREQ e di AUDIO. Il comando del volume dell'amplificatore audio verrà ruotato vicino al minimo e quello di POT1 all'incirca a metà della sua corsa. Collegando l'alimentazione si deve udire un suono sull'amplificatore il cui volume si può regolare a volontà del lettore; potremo modificare la frequenza premendo su FREQ UP e osservando che cambia anche il LED illuminato e il suono.

Di seguito ripeteremo la prova premendo P1 o P2, facendo funzionare un filtro passabasso che taglia le frequenze più alte del segnale.

Con il potenziometro POT1 si può modificare la frequenza di taglio del filtro.

LISTA DEI COMPONENTI

U1	Circuito integrato 4001
R1	Resistenza 4K7 (giallo, viola, rosso)
R2	Resistenza 100 K (marrone, nero, giallo)
C1,C3	Condensatore 100 nF
C2	Condensatore 22 μF

Programmazione con il PICBASIC PLUS LITE

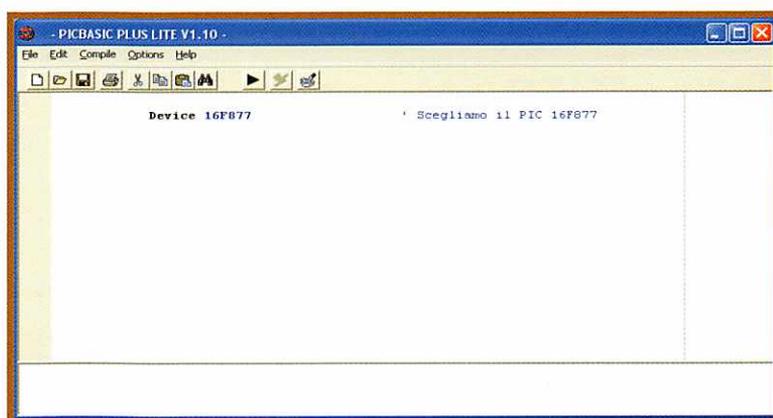
Per iniziare a programmare con il PicBasic Plus Lite dobbiamo analizzare la struttura generale che ha il codice realizzato con questo programma e conoscere le parole chiave. Presto potrete realizzare i vostri primi programmi in questo ambiente di programmazione.

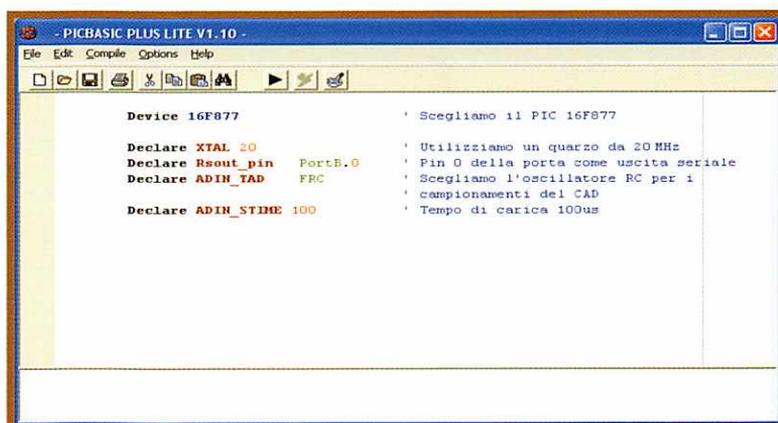
Ordine del programma

Dobbiamo abituarci a mantenere lo stesso ordine nella costruzione di un codice. Così come avviene nella programmazione in assembler tutte le varie parti o categorie di programmazione devono mantenere un particolare ordine. Questo sarà definito dal compilatore che ha un suo senso logico all'interno delle tecniche di programmazione.Tutti i programmi devono seguire la struttura riportata nella tabella sottostante.

La prima cosa da fare è definire il PIC che vogliamo utilizzare. È necessario definirlo all'inizio del codice in modo che il compilatore possa interpretare l'informazione successiva all'interno del programma. Grazie all'istruzione "DEVICE" possiamo scegliere tra i modelli 16F84 e 16F877, dato che questa versione del software accetta solo PIC con architettura da 14 bit ed è limitata solamente a questi due modelli. Potremo applicare tutto ciò che abbiamo imparato per il PIC16F870, dato che il PIC16F877 è della stessa famiglia e riunisce tutte le caratteristiche che sono state spiegate per il PIC16F870.

DEVICE	{tipo PIC}
DECLARE	{dispositivo} {valore}
DIM	{variabili}
SYMBOL	{nome} = {porta.pin}
DEFINE	{porta} = {input/output}
DATA	{tabelle} {...istruzioni di programma}
END	

Struttura di un programma in PicBasic Plus Lite.

Definizione del dispositivo tramite l'istruzione "DEVICE".

Dopo aver dichiarato il PIC che utilizzeremo possiamo definire i dispositivi esterni che verranno utilizzati nel programma. Quindi, mediante l'istruzione "DECLARE" assegniamo le variabili che sono necessarie per utilizzare dispositivi quali il display LCD, una tastiera, il bus I2C, la memoria dei dati EEPROM, la comunicazione seriale e i convertitori CAD. Sempre mediante questa istruzione dovremo defi-

Definizione degli elementi esterni "DECLARE".

Esempio di utilizzo dell'istruzione "DIM".

nire la frequenza di funzionamento, ovvero il quarzo che utilizzeremo all'interno del circuito elettronico. Questa versione del software, oltre alla limitazione delle 20 linee di codice, ha una limitazione riguardo alle frequenze, dato che permette di lavorare solamente con due frequenze: 4 e 20 MHz. Nella figura possiamo vedere un esempio di come utilizzare questa istruzione in un programma che utilizza un convertitore analogico-digitale (CAD).

Parole riservate che non potranno essere utilizzate come nomi riservati				
PP0	PP2H	PP5	PP7H	GEN4
PP0H	PP3	PP5H	GEN	GEN4H
PP1	PP3H	PP6	GENH	GPR
PP1H	PP4	PP6H	GEN2	BPF
PP2	PP4H	PP7	GEN2H	

Parole riservate.

Diversi modi di utilizzare l'istruzione "SYMBOL".

Definiamo i pin delle porte come I/O.

Con l'istruzione "DIM" si dichiarano le variabili del programma. Queste possono essere solamente bit, byte (8 bit) o word (16 bit). È obbligatorio dichiarare tutte le variabili prima di iniziare il loro utilizzo, dato che in caso contrario il compilatore ci darà un errore. Questa istruzione serve anche per delle variabili utilizzando dei nomi diversi, cioè potremo fare riferimento alla variabile utilizzando il suo nome oppure utilizzando un nome di comodo. Esistono una serie di parole riservate che non potranno essere utilizzate come nomi di variabili, riportate nella tabella della pagina precedente.

In ultimo, l'istruzione "DIM" può anche essere utilizzata per creare delle costanti, specificando il nome di quest'ultima e il valore fisso che avrà nel programma. Nella figura della pagina precedente è riportato un esempio di utilizzo fisso di istruzione per i diversi casi presentati. L'istruzione "SYMBOL" permette di dare un nome simbolico ai pin delle porte di ingresso/uscita. Questo nome normalmente viene assegnato in base all'utilizzo che si vorrà fare di questi pin e contribuisce a rendere più semplice la programmazione. Il nome può essere uno qualsiasi, la porta deve essere A, B o C, e il pin è il numero di piedino all'interno della stessa porta. Ogni riferimento successivo a questo pin all'interno del programma potrà essere fatto utilizzando unicamente il nome che gli abbiamo assegnato.

Questa istruzione può anche essere utilizzata per dare un nome diverso a una variabile o a una costante, ma non può essere utilizzata per creare una variabile. Se creiamo una costante con questa istruzione non si potrà utilizzare la memoria RAM per contenerla.

Nella figura possiamo vedere diversi esempi dell'utilizzo dell'istruzione "SYMBOL".

"DEFINE" si utilizza per definire tutti i pin delle porte come ingresso o uscita. Di solito questa istruzione non si utilizza, dato che il compilatore permette di fare direttamente l'assegnazione. Ricordate che per definire i pin delle porte come ingressi o uscite, dovremo configurare i registri TRISX associati alle porte. Con il valore 1 configureremo un pin come ingresso e con il valore 0 come uscita.

Mediante l'istruzione "DATA" possiamo definire una tabella di dati alfanumerici, cioè una serie di caratteri alfabetici e/o numerici. Ognuno dei caratteri definiti è un dato a cui si può fare accesso per leggerlo. I caratteri alfabetici devono sempre essere contenuti fra virgolette

```
- PICBASIC PLUS LITE V1.10 - D:\PBPLUS_LITE\File per ED\ed65.bas
File Edit Compile Options Help

Device 16F877                      ' Scegliamo il PIC 16F877
Declare XTAL 20                    ' Utilizziamo un quarzo da 20 MHz
DIM Gatto as BYTE                  ' Crea una variabile da 8 bit (0-255)
SYMBOL LED = PORTA.1               ' LED riferito al bit-1 della PortA
Define TrisB.0 = 1                 ' bit-0 della PortB come ingresso
TRISB.1 = 0                        ' bit-1 della PortB come uscita
Data "a","b","c",1,3,"Maria"       ' Creiamo una tabella di dati
```

Creiamo una tabella con l'istruzione "DATA".

(""), diversamente il compilatore interpreterà questo carattere come il nome di una variabile. Non tutte le istruzioni che abbiamo analizzato sono utilizzate insieme nello stesso programma, si utilizzano solamente quelle che sono strettamente necessarie. Mantenendo l'ordine logico nella programmazione, dopo aver preparato il "materiale" per lavorare, continuiamo con l'inserimento delle istruzioni del programma. Con queste istruzioni diciamo al PIC ciò che vogliamo fargli fare, le istruzioni che si utilizzano nel PicBasic Plus Lite, si possono trovare all'interno dell'help del programma stesso, nel documento in formato ".pdf" che si trova nella stessa cartella del primo CD, in cui è contenuto il programma.

Per dichiarare di aver terminato il codice dobbiamo inserire l'istruzione "END".

In questo modo il compilatore capirà che il codice è terminato e non compilerà nessun'altra istruzione.

Possiamo utilizzare etichette all'interno del codice.

```
- PICBASIC PLUS LITE V1.10 - D:\PBPLUS_LITE\File per ED\ed65.bas
File  Edit  Compile  Options  Help

         Device 16F877                   ' Scegliamo il PIC 16F877
         Declare XTAL 20                 ' Utilizziamo un quarzo da 20MHz
         DIM Gatto as BYTE               ' Crea una variabile da 8 bit (0-255)
         SYMBOL LED = PORTA.1            ' LED riferito al bit-1 della PortA
         Define TrisB.0 = 1              ' bit-0 della PortB come ingresso
         TRISB.1 = 0                     ' bit-1 della PortB come uscita
         Data "a","b","c",1,3,"Maria"    ' Creiamo una tabella di dati
Inizio:  REM Il programma inizia qui
         REM istruzioni
         REM istruzioni
         end
```

Un altro modo di inserire commenti, "REM".

Etichette

L'utente può definire ciò che conosciamo come "etichette" ed evitare così di fare riferimento a indirizzi di memoria, quando deve eseguire dei file. Le etichette devono essere all'inizio della linea, e devono essere seguite da due punti (:) e uno spazio.

Commenti

Nelle figure presentate come esempi avete potuto verificare che la spiegazione di ciò che esegue ogni linea è inserita alla destra della linea stessa, preceduta da una virgoletta semplice ('). Dobbiamo utilizzare i commenti per rendere più facile l'interpretazione del codice. Un altro modo di definire un commento è l'utilizzo dell'istruzione "REM". Tutto ciò che sulla stessa linea arriva dopo questa istruzione verrà interpretato dal compilatore come un commento.

Tabulazioni

È consigliabile che le linee di programma a eccezione delle etichette, siano tabulati. In questo modo otterremo anche che il codice si possa comprendere con maggiore facilità.

Mantenendo sempre un ordine e gli stessi criteri durante la programmazione, questo lavoro risulterà molto più semplice e ottimizzerà i tempi di sviluppo di una applicazione.

Esercizio: visualizzazione di messaggi, il programma

Continueremo a esercitarci con il display LCD mediante un nuovo esercizio. Aumenteremo progressivamente la difficoltà delle applicazioni, acquisendo in questo modo maggiori conoscenze e dimestichezza, con questo dispositivo e con il PC in generale.

Enunciato

Si tratta di sviluppare un'applicazione che permetta di visualizzare sul modulo LCD diversi messaggi. Sul display verranno presentati i messaggi "Ciao" e "Addio" alternativamente, con un intervallo di tempo di 2 secondi.

Dall'enunciato possiamo dedurre i dispositivi del microcontroller che utilizzeremo. Avremo bisogno delle porte A e B per lavorare con LCD e del Timer0 per stabilire la temporizzazione.

Organigramma

Prima di iniziare a progettare il codice, lo dovremo plasmare impostandolo attraverso l'organigramma. La prima cosa da prendere in considerazione è la necessità di configurare i dispositivi del PIC.

Fatto questo, inizializzeremo il display LCD e lo configureremo (abilitazione, cursore e lampeggio) preparando il modulo per lavorare con esso. Visualizzeremo il primo messaggio ed eseguiremo un chiamata alla subroutine di temporizzazione caricandola con il valore che provoca il ritardo desiderato. Trascorso questo tempo, visualizzeremo il secondo messaggio e temporizzeremo nuovamente, prima di ripete-

Organigramma dell'applicazione.

Ogni due secondi cambierà il messaggio sul display LED.

addio.asm - Blocco note

File Modifica Formato Visualizza ?

```
;       ESERCIZIO: LCD con diversi messaggi
;       ----------------------------------
;

                List    p=16F870            ;Tipo di processore
                include "P16F870.INC"       ;Definizione dei registri interni

Lcd_var         equ     0x20                ;Variabili (2) delle routine di gestione del LCD
Delay_Cont      equ     0x21                ;Variabile per la temporizzazione
Temporale_1     equ     0x22                ;Variabile Temporale
Temporale_2     equ     0x23                ;Variabile Temporale

                org     0x00                ;Vector di Reset
                goto    Inizio
                org     0x05                ;Salva vector di interrupt

                include "lcd_cxx.inc"       ;Include la routine di gestione del LCD

;**************************************************************************************
```

Intestazione del programma.

re il ciclo dal punto in cui visualizzeremo il primo messaggio.

Codice Intestazione del programma

Come d'abitudine, la prima cosa da fare quando apriamo l'editor di testo per creare un nuovo programma, è scrivere all'interno di un commento la spiegazione di ciò che vogliamo fare in un programma. Definiamo il microcontroller che utilizzeremo e la libreria di definizione dei registri. Quando lavoriamo con il display LCD dobbiamo inserire anche la libreria che contiene le subroutine di funzionamento del display.

Terminiamo la prima fase del programma con le direttive ORG per organizzare il codice nella memoria e definendo le variabili che interverranno all'interno del programma.

Sono necessarie le due variabili per lavorare con la libreria dell' LCD, saranno definite come "Lcd_var", dato che nella libreria a questo nome corrisponde un blocco di due variabili. Definiremo la variabile che interverrà nella temporizzazione "Delay_Cont" e in ultimo, verranno definite le due variabili temporali per lo sviluppo del programma "Temporale_1" e "Temporale_2".

Nel caso non si conoscano le variabili necessarie per lo sviluppo del programma, verranno definite man mano che si rendono necessarie.

addio.asm - Blocco note

File Modifica Formato Visualizza ?

```
;**************************************************************************************
;Delay_var: Questa routine di carattere generale realizza una temporizzazione variabile
;fra 50 ms e 12.8". Si utilizza un prescaler da 256 e si carica il TMR0 con 195.
;La velocità di lavoro è di 4 MHz quindi il TMR0 si incrementa ogni µs. In questo
;modo il TMR0 deve contare 195 eventi che, con un prescaler da 256 generano un
;intervallo totale di  50000 µs cioè 50 ms (195 * 256). Il valore 195 deve essere espresso
;in Hex. (c3) e dato che il TMR0 è ascendente dovremo caricare il suo complemento (3C hex.)
;Questo intervallo di 50 ms è ripetuto tante volte quante indicate dalla variabile "Delay_cont",
;è per questo che il ritardo minimo è 50 ms ("Delay_cont=1) e il massimo è 12.8"
;(Delay_cont=255).

Delay_var:      bcf     INTCON,T0IF         ;Resetta il flag di overflow
                movlw   0x3c                ;Complemento hex. di 195
                movwf   TMR0                ;Carica il TMR0
Intervallo      clrwdt                      ;Aggiorna il WDT
                btfss   INTCON,T0IF         ;Overflow del TMR0?
                goto    Intervallo          ;Non ancora
                decfsz  Delay_Cont,F        ;Decrementa il contatore di intervalli
                goto    Delay_var           ;Ripete l'intervallo di 50 ms
                return

;**************************************************************************************
```

Routine di temporizzazione.

addio.asm - Blocco note

File Modifica Formato Visualizza ?

```
;*****************************************************************************************************
;In base al valore contenuto nel registro W, si ottiene il carattere da visualizzare

Tab_Messaggi    movwf   PCL             ;Calcola lo spostamento sulla tabella

Mess_0          equ     $               ;Mess_0 punta al primo carattere del messaggio 0
                retlw   'C'
                retlw   'i'
                retlw   'a'
                retlw   'o'
                retlw   0x00            ;Ultimo carattere del messaggio 0

Mess_1          equ     $               ;Mess_1 punta al primo carattere del messaggio 1
                retlw   'A'
                retlw   'd'
                retlw   'd'
                retlw   'i'
                retlw   'o'
                retlw   0x00            ;Ultimo carattere del messaggio 1

;*****************************************************************************************************
```

Tabella dei messaggi.

addio.asm - Blocco note

File Modifica Formato Visualizza ?

```
;******************************************************************************
;
;
Inizio          clrf    PORTB           ;Cancella i latch di uscita
                bsf     STATUS,RP0      ;Seleziona banco 1
                clrf    TRISB           ;Configura la Porta B come uscita
                movlw   b'00011000'
                movwf   TRISA           ;RA0-RA2 uscite, RA3-RA4 ingressi
                movlw   b'00000110'
                movwf   ADCON1          ;Configuriamo le porte come uscite digitali
                movlw   b'00000111'
                movwf   OPTION_REG      ;Prescaler da 256 per il TMR0
                bcf     STATUS,RP0      ;Seleziona banco 0

                call    LCD_INI         ;Sequenza di Inizio del LCD
                movlw   b'00001100'
                call    LCD_REG         ;Invia istruzione: LCD ON, Cursore OFF e blink OFF
```

Configurazione dei dispositivi.

Routine di temporizzazione

Sappiamo che abbiamo bisogno di una routine di temporizzazione. Dato che negli esercizi precedenti abbiamo già utilizzato routine di ritardo o temporizzazioni, copieremo la routine nel nostro codice.

Nella figura alla pagina precedente possiamo vedere una routine di temporizzazione tipica degli esercizi che abbiamo eseguito fino a questo momento.

Tabella dei messaggi

Dato che presenteremo due messaggi diversi ('Ciao' e 'Addio') e che questi devono essere inviati carattere per carattere al modulo LCD, creeremo una tabella dove contempleremo tutti i caratteri. Quando richiameremo la tabella , il PCL o contatore di programma si caricherà con il valore del registro di lavoro e in funzione di quest'ultimo restituirà il carattere da visualizzare. Quando il valore fornito è 0 significa che il messaggio è stato presentato nella sua completezza. Nell'immagine della figura in alto potete vedere il formato di questa tabella.

Configurazione dei dispositivi

Senza aver iniziato a programmare abbiamo già dei blocchi di codice nel nostro file. Inizieremo il programma principale che risolve l'applicazione interagendo con i blocchi e le librerie incluse, configurando i dispositivi del PIC che utilizzeremo.

Definiamo la porta B come porta di uscita e nella porta A imposteremo RA0-RA2 come uscite e RA3-RA4 come ingressi. Configurando il registro ADCON1 definiremo la porta A di tipo digitale e configurando il registro OPTION_REG definiremo il predivisore per il temporizzatore Timer0. Infine inizializzeremo il display e lo configureremo in modo che sia abilitato, con cursore e lampeggio definiti.

ADIOS - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
;*******************************************************************************
;Mensaje: Esta rutina visualiza en el LCD el mensaje cuyo inicio está  indicado en
;el acumulador. El fin de un mensaje se determina mediante el código 0x00

Mensaje         movwf   Temporal_1       ;Salva posición de la tabla
Mensaje_1       movf    Temporal_1,W     ;Recupera posición de la tabla
                call    Tabla_Mensajes   ;Busca caracter de salida
                movwf   Temporal_2       ;Guarda el caracter
                movf    Temporal_2,F
                btfss   STATUS,Z         ;Mira si es el último
                goto    No_es_ultimo
                return
No_es_ultimo    call    LCD_DATO         ;Visualiza en el LCD
                incf    Temporal_1,F     ;Siguiente caracter
                goto    Mensaje_1

;*******************************************************************************
```

Subroutine per la configurazione del messaggio.

addio.asm - Blocco note

File Modifica Formato Visualizza ?

```
Loop        movlw   b'00000001'
            call    LCD_REG          ;Cancella LCD e Home (colloca il cursore nella 1ª posizione)
            movlw   Mess_0
            call    Messaggio        ;Visualizza il Messaggio 0
            movlw   .20
            movwf   Delay_Cont
            call    Delay_var        ;Temporizza 2 secondi
            movlw   b'00000001'
            call    LCD_REG          ;Cancella LCD y Home (colloca il cursore nella 1ª posizione)
            movlw   Mess_1
            call    Messaggio        ;Visualizza il Messaggio 1
            movlw   .20
            movwf   Delay_Cont
            call    Delay_var        ;Temporizza 2 secondi
            goto    Loop

            end                      ;Fine del programma sorgente
```

Programma principale.

Programma principale

Questa è la parte del codice in cui risolveremo realmente l'applicazione. La prima cosa da fare è inviare un comando al modulo LCD per posizionare il cursore nella posizione iniziale del display. Ora dobbiamo decidere come visualizzare i messaggi.

Per vedere i messaggi dobbiamo prima sapere quale dobbiamo presentare sul display e dopo muoverci sulla tabella presentando carattere per carattere fino ad arrivare all'ultimo. Per eseguire questo creeremo una routine che visualizzi sull'LCD il messaggio il cui inizio è indicato nell'accumulatore e che identifichi la fine di un messaggio determinato mediante il codice 0x00.

Questa routine ha bisogno di due variabili, una per salvare l'indirizzo della tabella da cui abbiamo ottenuto il carattere e l'altra per salvare il carattere stesso. Queste variabili sono Temporale_1 e Temporale_2.

La prima cosa che fa la routine è salvare l'indirizzo della tabella nella prima variabile temporale. Dopo inizia un ciclo in cui si scrive nella variabile l'indirizzo del carattere da visualizzare, si cerca nella tabella e si scrive nella seconda variabile. Nel ciclo si verifica anche che questo carattere non sia l'ultimo, e nel caso in cui non lo sia, lo si visualizza chiamando la routine LCD_DATO e si incrementa la posizione della tabella per andare al carattere successivo. Se è l'ultimo si esce dal ciclo e dalla routine.

Abbiamo risolto in una subroutine come presentare i messaggi, dato che il programma principale richiamerà questa subroutine, caricherà la variabile di temporizzazione con il valore appropriato per il ritardo desiderato e richiamerà la subroutine di ritardo. Fatto questo, ripeteremo gli stessi passaggi per il messaggio successivo: posizionamento del cursore, routine dei messaggi e routine di ritardo.

Tutto questo sarà all'interno di un ciclo per ripetere ciclicamente questi passaggi.

Compilazione

Abbiamo terminato quella che si definisce la prima stesura del programma. È normale, durante la compilazione, trovare qualche errore che non ci costerà molta fatica risolvere. L'esempio che abbiamo presentato si trova sul secondo CD allegato all'opera, all'interno della cartella "LCD" con il nome "Addio.asm".

Quando apriamo MPLAB ed eseguiamo tutti i passaggi necessari per compilare il programma vedremo che la compilazione non genererà errori, ma darà i messaggi tipici di posizionamento dei registri nella memoria.

Build Results

```
Building ADIOS.HEX...

Compiling ADIOS.ASM:
Command line: "C:\ARCHIV~1\MPLAB\MPASMWIN.EXE /p16F870 /q C:\ARCHIV~1\MPLAB\PROYEC~1\ADIOS.ASM"
Message[302] C:\ARCHIV~1\MPLAB\PROYEC~1\ADIOS.ASM 84 : Register in operand not in bank 0.  Ensure
Message[302] C:\ARCHIV~1\MPLAB\PROYEC~1\ADIOS.ASM 86 : Register in operand not in bank 0.  Ensure
Message[302] C:\ARCHIV~1\MPLAB\PROYEC~1\ADIOS.ASM 88 : Register in operand not in bank 0.  Ensure

Build completed successfully.
```

Risultato dell'applicazione.